Architecture Analysis with AADL
The Speed Regulation Case-Study

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Julien Delange

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **NOV 2014**	2. REPORT TYPE	3. DATES COVERED **00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE **Architecture Analysis with AADLThe Speed Regulation Case-Study**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**56**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

DM-0001524

What this talk is about?

1. Actual issues for Safety-Critical systems design

2. Why Model-Based Engineering techniques are helpful

3. How AADL can detect issues early and avoid potential rework

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

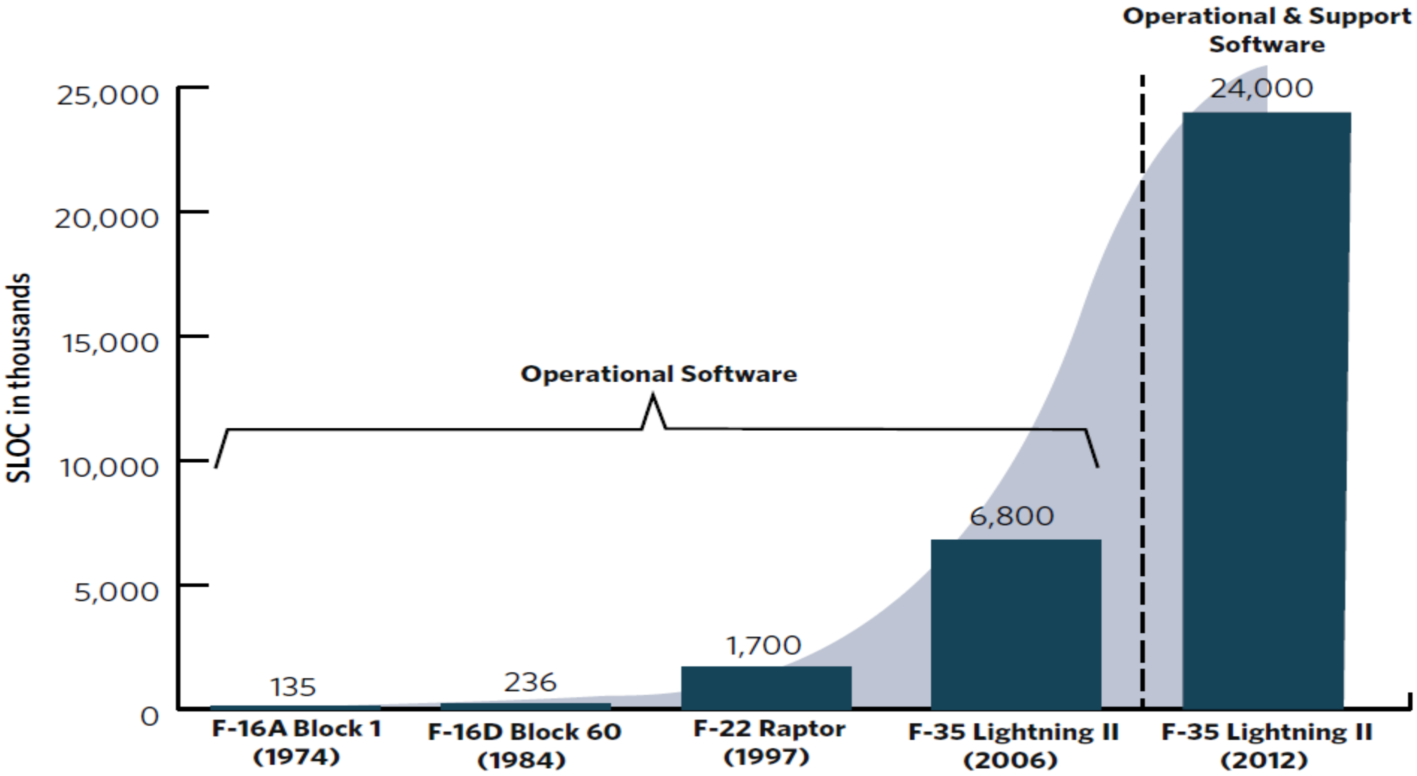
Architecture Analysis

Conclusion

Polling Question 1

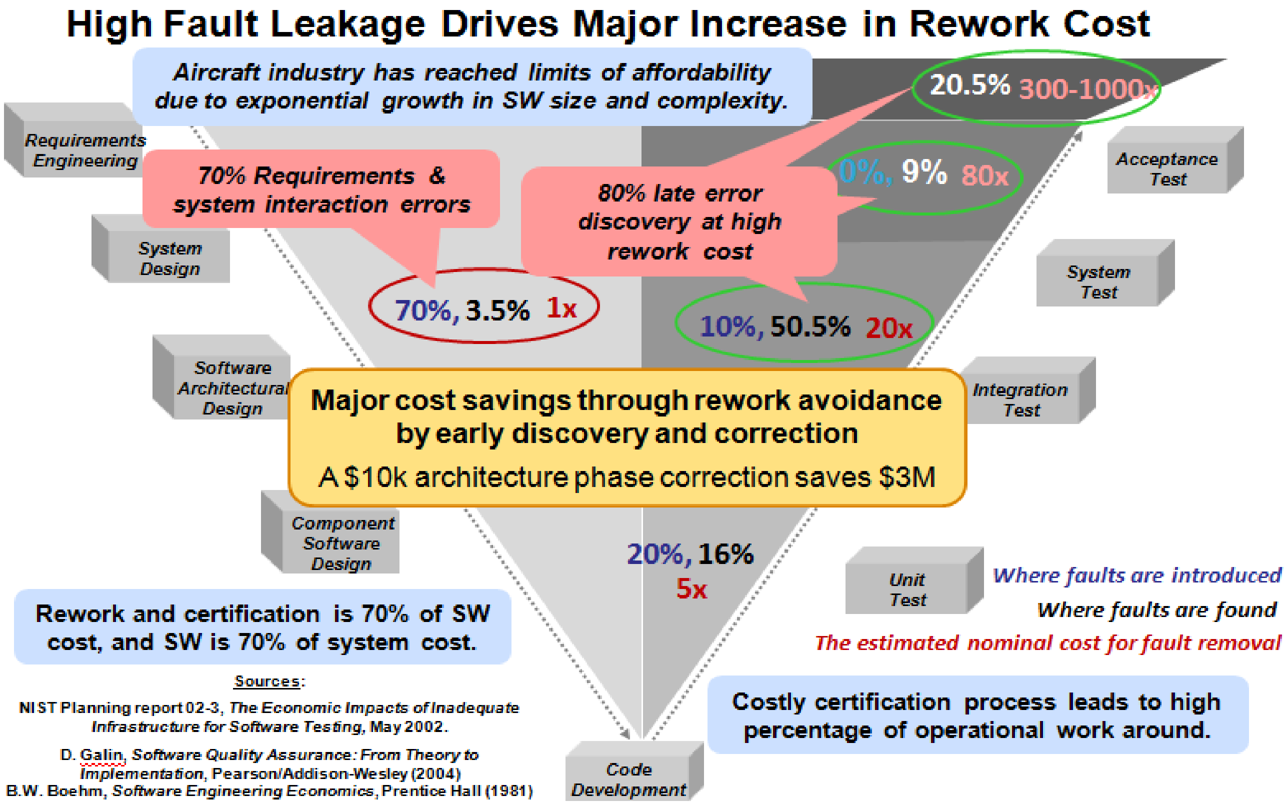
Do you know what Model-Based Engineering is?

Safety-Critical Systems are Intensively Software-Reliant



Source: *"Delivering Military Software Affordably"* in Defense AT&L

Errors are introduced early but detected (too) lately



Sources:

NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.

D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)

B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)

Many Errors stems from Architecture or Integrati... ...ues

Global Variable used among different functions

Potential issues: inconsistent values, ...

Root Cause: Architecture Desi...

Use of COTS components with...

Potential impa...

Root Ca... ...gration

Timing is...

... enforced, bad values

... ...ation policy, lack of analysis

...ould I continue this list?

Fact1: All these errors could be detected at Design-Time

Fact2: They are actually detected during integration tests

Fact3: They incur rework costs and postpone product delivery

Why Model-Based Engineering Matters?

Capture system architecture with designers requirements

- Focus on system structure/organization (e.g. shared components)
- Tailor architecture to specific engineering domain (e.g. safety)

Validate the architecture

- Check requirements enforcement (e.g. no global variable)
- Detect Potential issues (e.g. interfaces consistency)

Early Analysis

- Avoid late re-engineering efforts (e.g. less rework after integration)
- Support decisions between different architecture variations

Polling Question 2

Do you already know AADL?

Architecture Analysis Design Language

SAE Standard for Model-Based Engineering

First version in 2003, actual version 2.1

Definition of System and Software Architecture

Specialized components with interfaces (not just “blocks”)

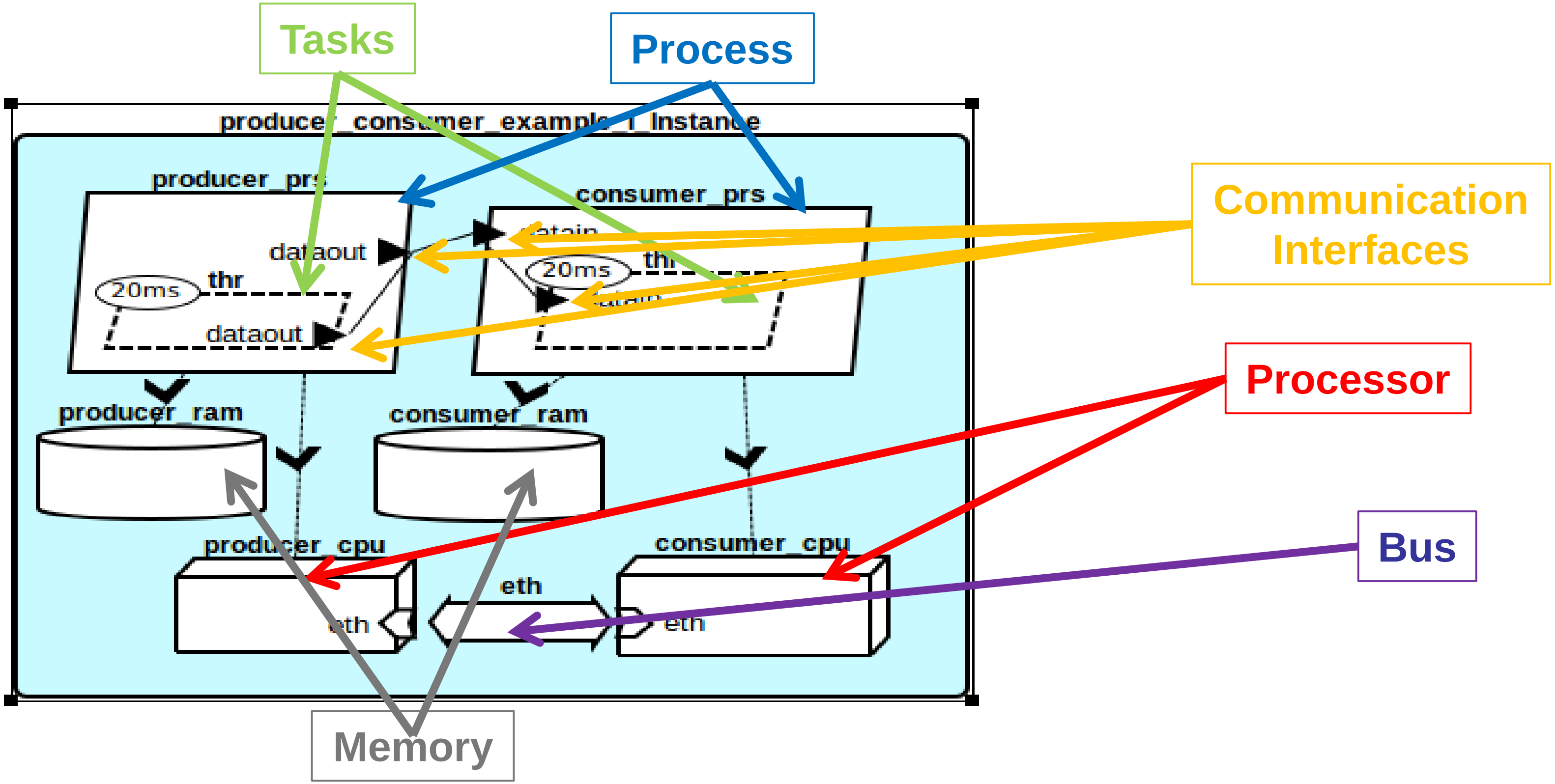
Interaction with the Execution Environment (processor, buses)

Extension mechanisms

User-Defined Properties (integrate your own constraints)

Annexes (existing for safety, behavior, etc.)

AADL Model Example



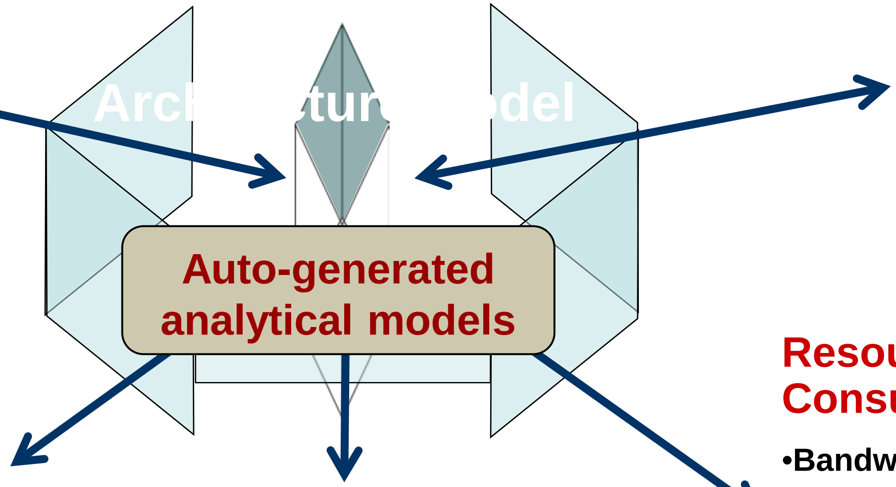
Architecture Analysis Design Language

Safety & Reliability

- MTBF
- FMEA
- Hazard analysis

Security

- Intrusion
- Integrity
- Confidentiality



Data Quality

- Data precision/ accuracy
- Temporal correctness
- Confidence

Real-time Performance

- Execution time/ Deadline
- Deadlock/starvation
- Latency

Resource Consumption

- Bandwidth
- CPU time
- Power consumption

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion

Objectives of this Study

Learn Architecture Modelling with AADL and the OSATE workbench

Model a family of systems with their variability factors

Analyze the Architecture from a performance perspective

Discover Safety Issues using Architecture Models

Support Architecture Alternatives Selection

Illustrate the Process with a relevant case study

Case-Study Description

Self-Driving car speed regulation

Obstacle detection with user warning

- Camera detection
- Infra-red sensor

Automatic Speed and Brake

- Two speed (wheel, laser) sensors
- Redundant GPS

Polling Question 3

On what aspect would you like to focus?

Case-Study Objectives

Help designers to choose the *best* Architecture

- Best reliability, avoid potential failure/error
- Meet timing and performance requirements

Analyze Architecture according to stakeholders criteria

- Try to analyze what really matters

Quantify architecture quality from different perspectives

- **Latency**
- **Resources and Budgets**
- **Safety/Reliability**

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

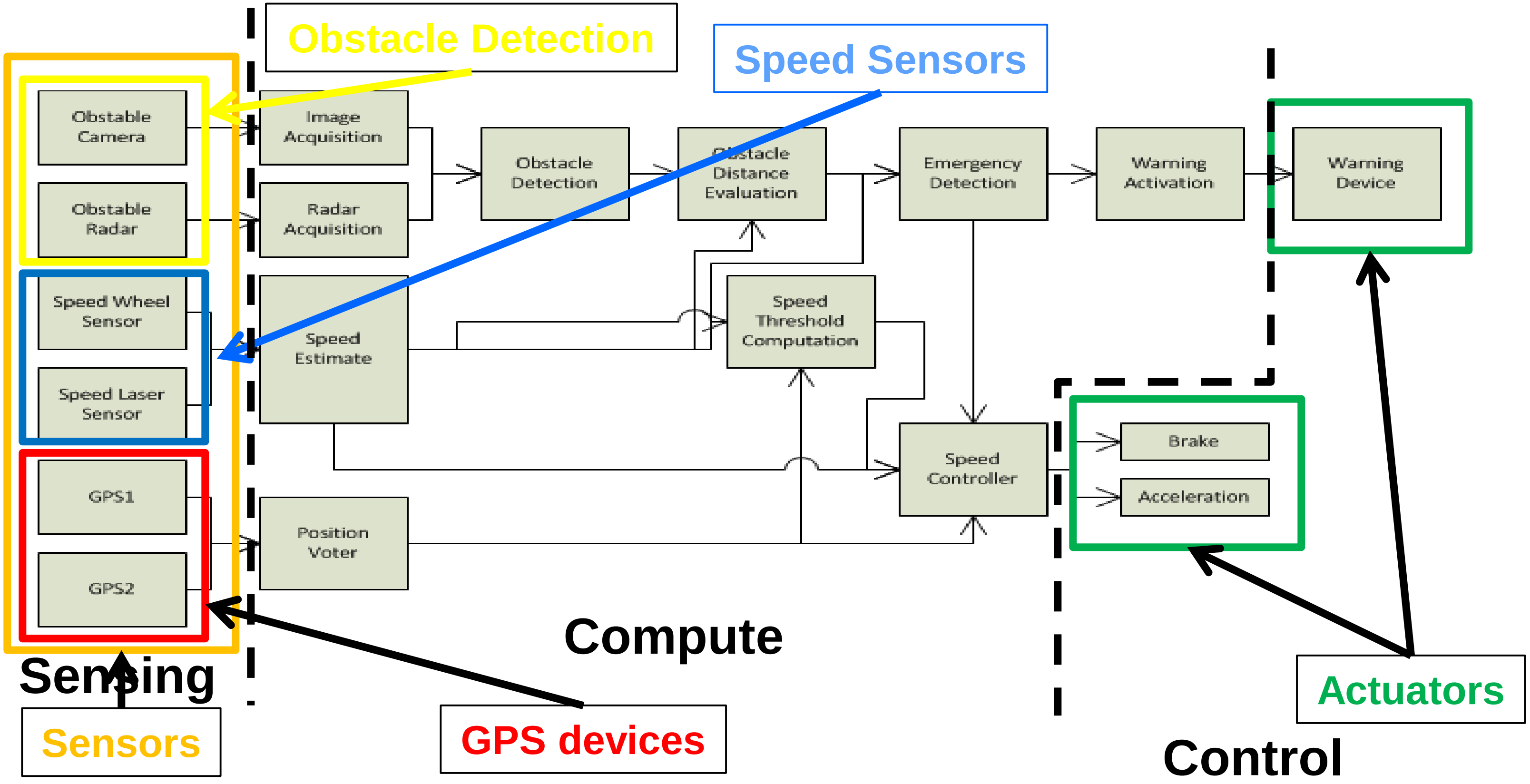
System Overview

AADL model description

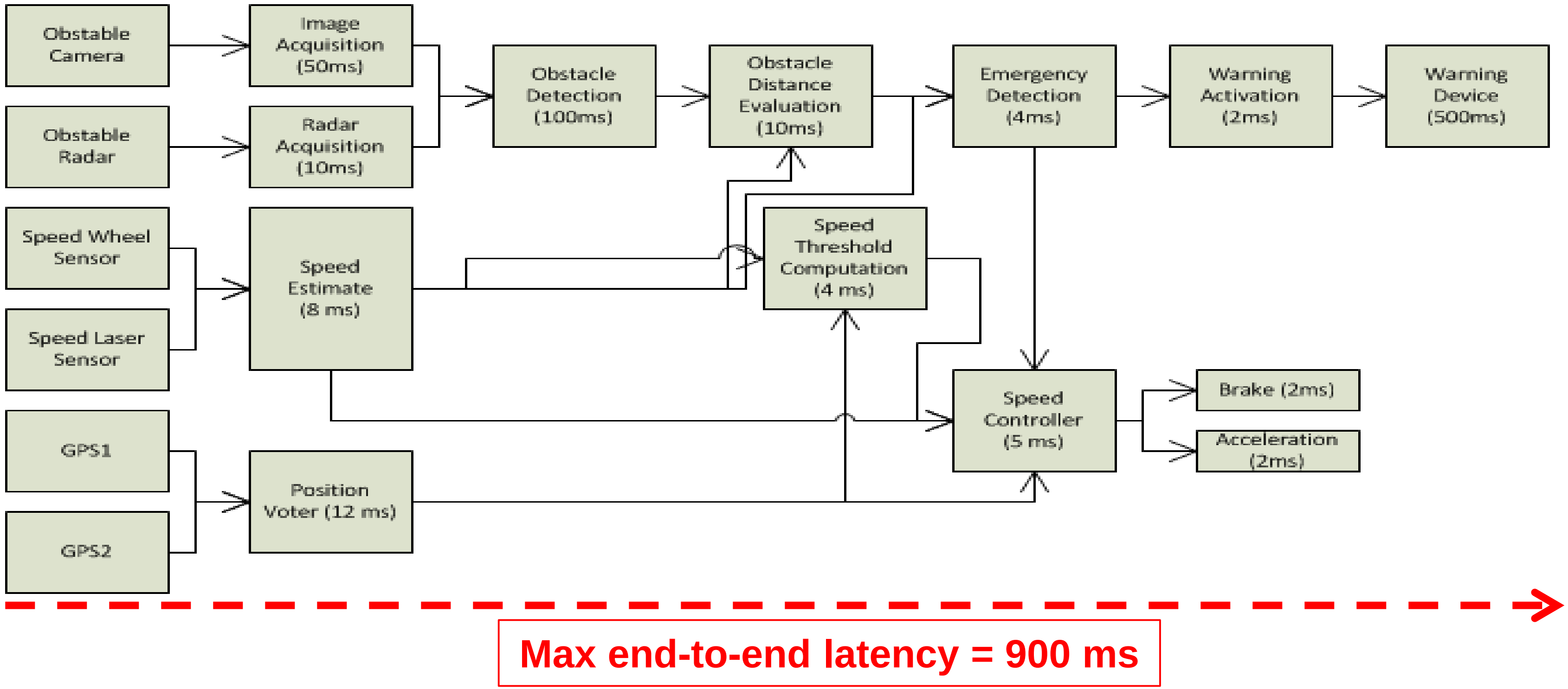
Architecture Analysis

Conclusion

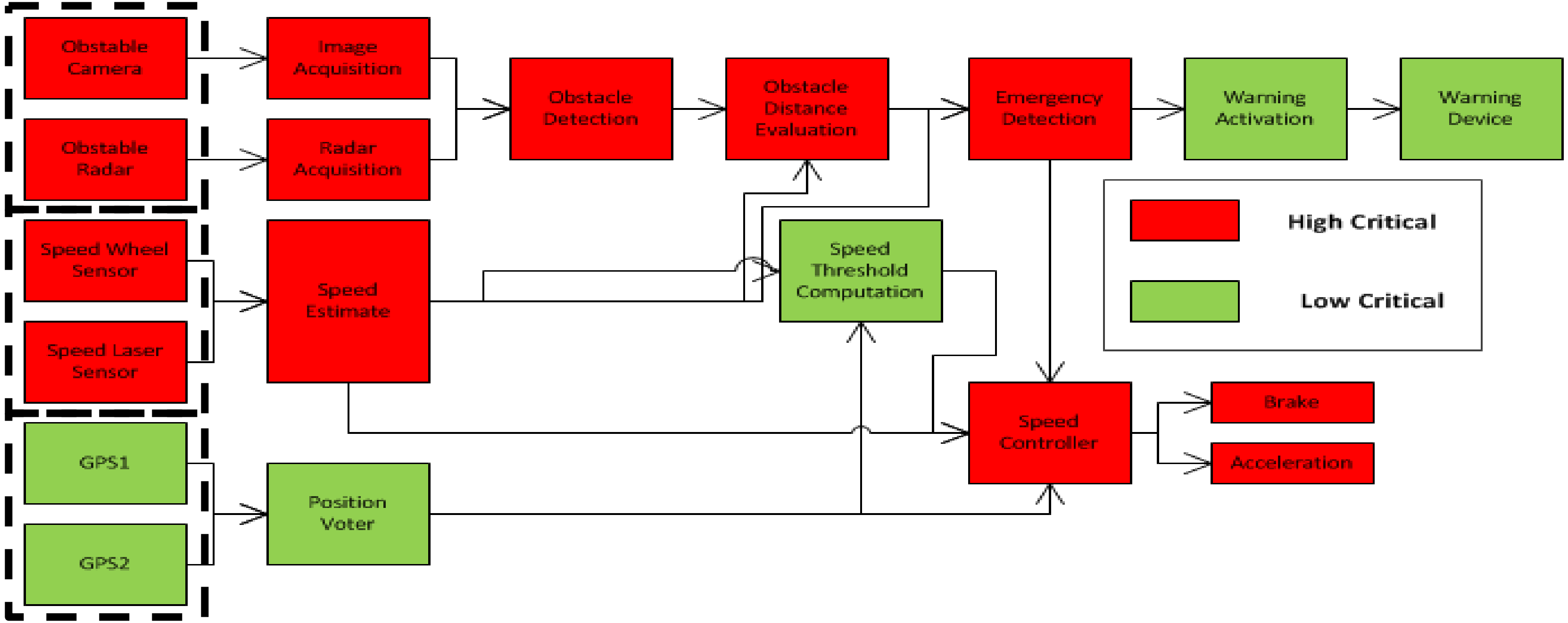
Functional Architecture



Functional Architecture, timing perspective



Functional Architecture, criticality perspective



Redundancy Groups (performs the same function)

Deployment Alternatives

Alternative 1: reduce cost and complexity

Two processors and one shared bus

Potential interactions for functions collocated on the same processor

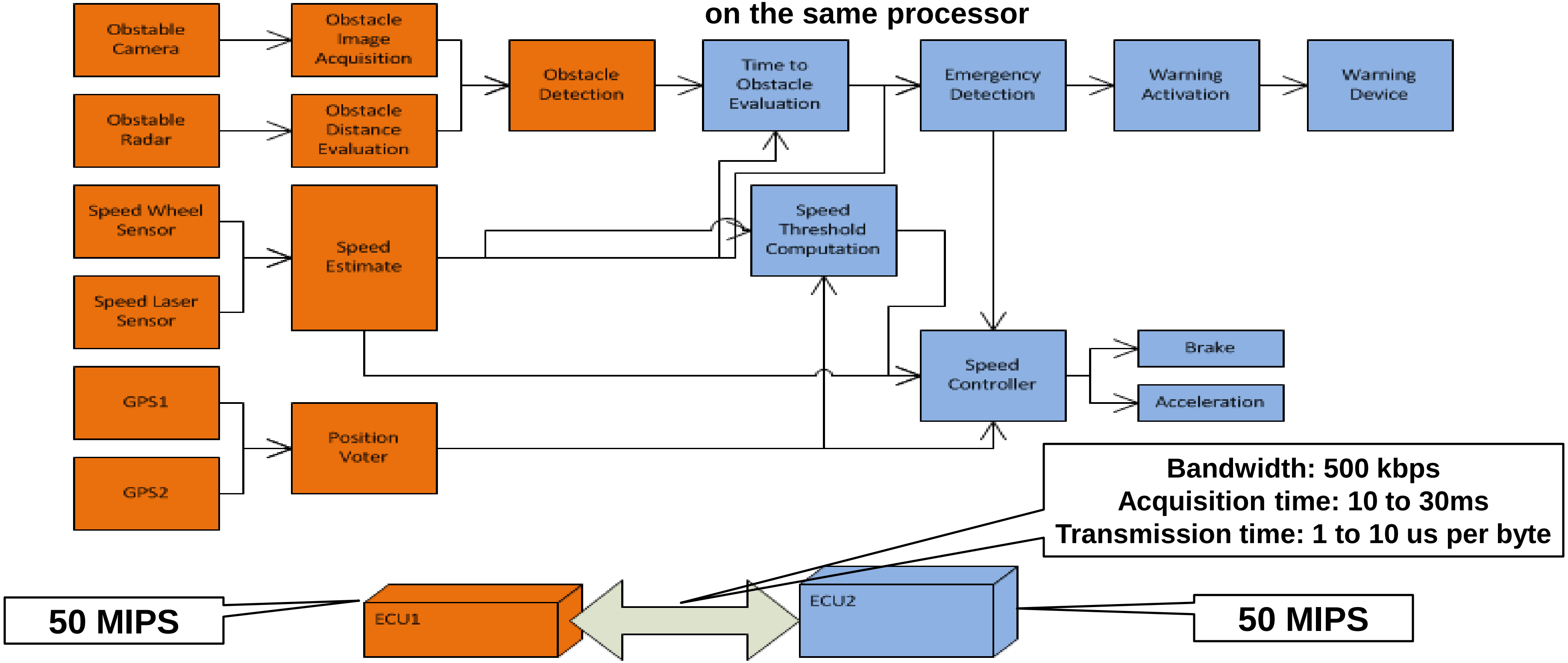
Alternative 2: reduce potential fault impact

Increase potential production cost (more hardware)

Three processors inter-connected with two buses

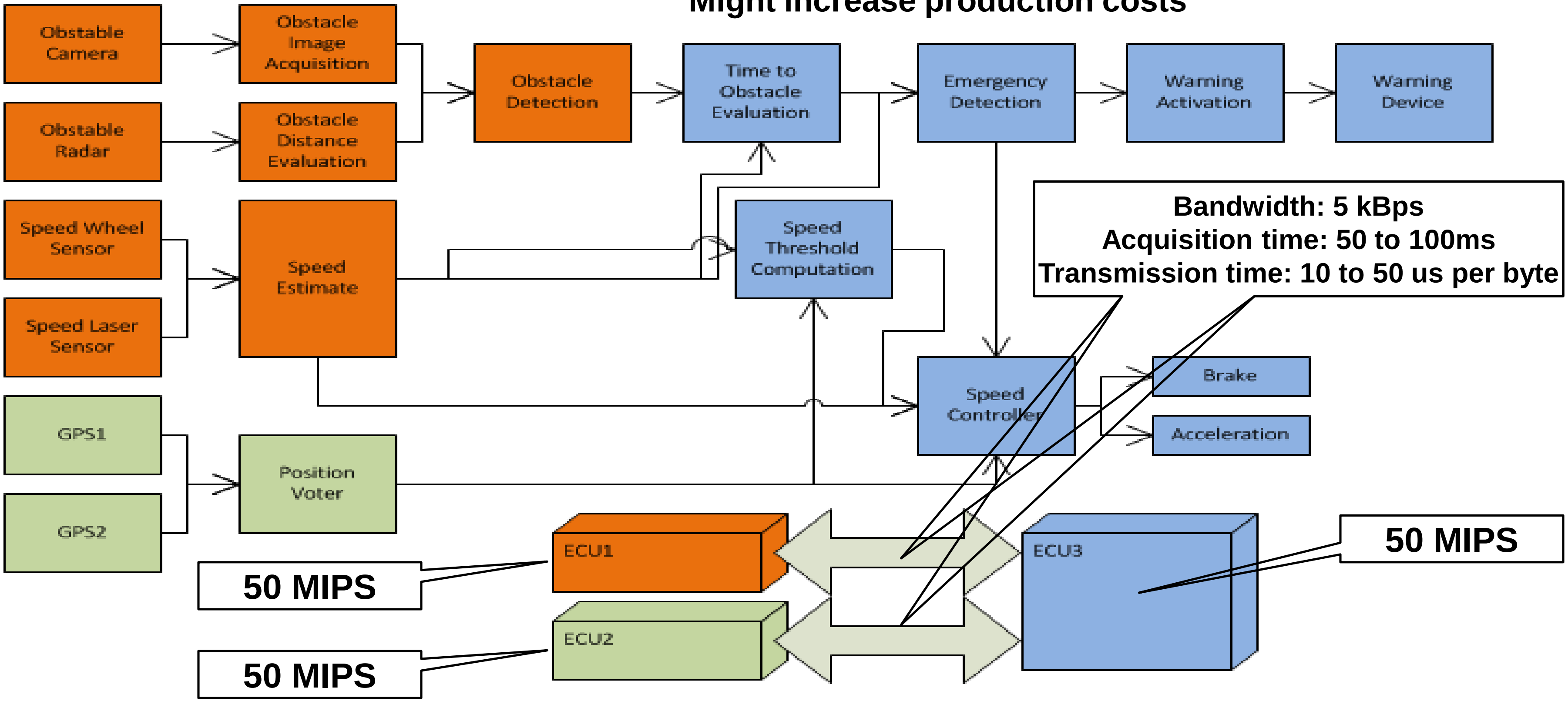
Architecture Alternative 1

Reduce Cost and Complexity
Potential interactions for functions collocated on the same processor



Architecture Alternative 2

Reduce Fault Impact
Might increase production costs



Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion

Modeling Guidelines

Separate architecture aspects in different files

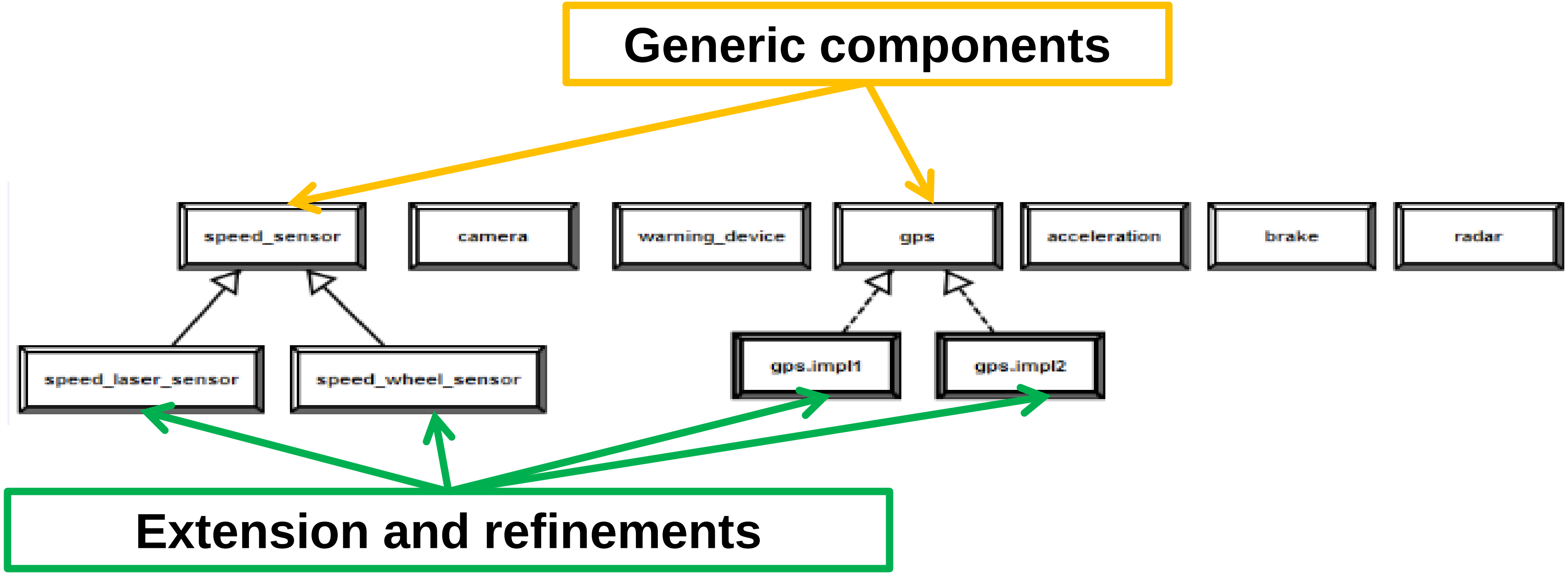
Leverage AADL extension and refinement mechanisms

- Capture common characteristics, avoid copy/paste
- Extend generic components

Use properties to quantify quality attributes

- Processed by tools to evaluate architecture quality
- **Specify once**, use by several analysis tools
- **Ensure Analyses Consistency**

Model Organization – devices



Model Organization – devices – textual model

```
device radar
features
    distance_estimate : out data port speed_regulation::icd::distance;
flows
    f0 : flow source di
properties
    Period => 10ms;
annex EMV2 {**
    use types speed_reg

    error propagations
        distance_estimate : out propagation {NoValue,InvalidValue};
    flows
        ef0 : error source distance_estimate{NoValue,InvalidValue};
    end propagations;

    properties
        emv2::severity => ARP4761::Major applies to distance_estimate.novalue;
        emv2::likelihood => ARP4761::Probable applies to distance_estimate.novalue;
        emv2::hazards =>
            ([  crossreference => "N/A";
                failure => "NoValue";
                phases => ("all");
                description => "No information from the Radar";
                comment => "Error if both the camera and the radar does not send any value";
            ])
            applies to distance_estimate.novalue;

        emv2::severity => ARP4761::Minor applies to distance_estimate.invalidvalue;
        emv2::likelihood => ARP4761::Probable applies to distance_estimate.invalidvalue;
        emv2::hazards =>
            ([  crossreference => "N/A";
                failure => "InvalidValue";
                phases => ("all");
                description => "Invalid distance sent by the radar";
                comment => "First occurrences of invalid data Should be handled by the distance estimator.";
            ])
            applies to distance_estimate.invalidvalue;
**};
end radar;
```

Component Name

Timing constraints
(latency analysis)

Error propagations and flows

Types of faults
(all safety analysis tools)

Documenting the faults

(safety analysis)

Model Organization – Interfaces Specifications

Data types being used to communicate across functions

Data size properties
(resource allocation and latency analysis)

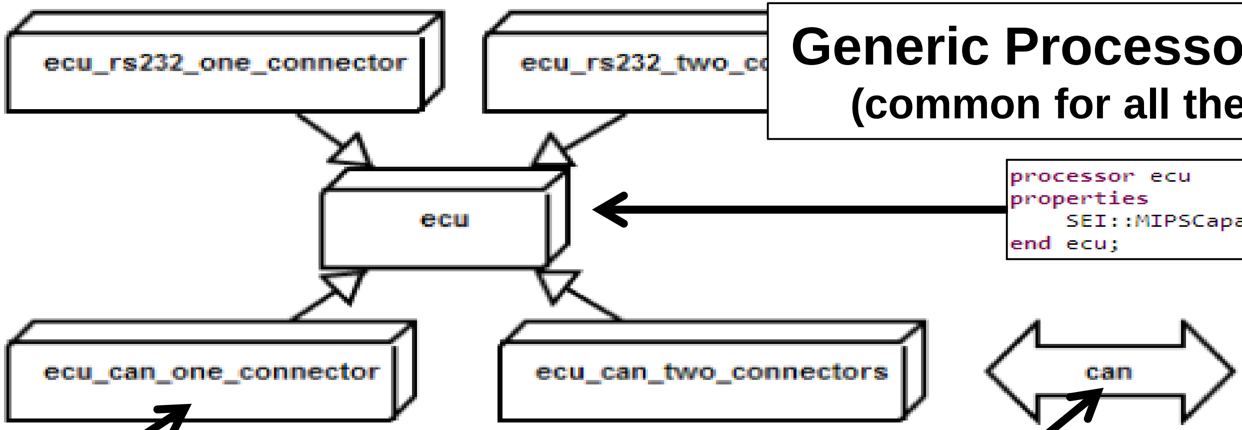
```
data gps_position
properties
    data_size => 50 Bytes;
    data_model::data_representation => Array;
end gps_position;

data sp
end

d
pr
    da
    data_mo
end picture;

data boolean
properties
    data_size => 1 bits;
end boolean;
```

One property, several analyses

⇨Ensure Analyses Consistency

```
  ta speed_command_type
properties
    ta_model::data_representation => enum;
     model::enumerators => ("brake", "accel");
    ize => 2 bits;
  command_type;

  speed_command
  nd speed_command;

data implementation speed_command.i
subcomponents
    kind : data speed_command_type;
    value : data base_types::unsigned_16;
end speed_command.i;

data distance extends base_types::unsigned_32
end distance;
```

Model Organization – platform



Generic Processor Component
(common for all the architecture)

```
processor ecu
properties
    SEI::MIPSCapacity => 50.0 MIPS;
end ecu;
```

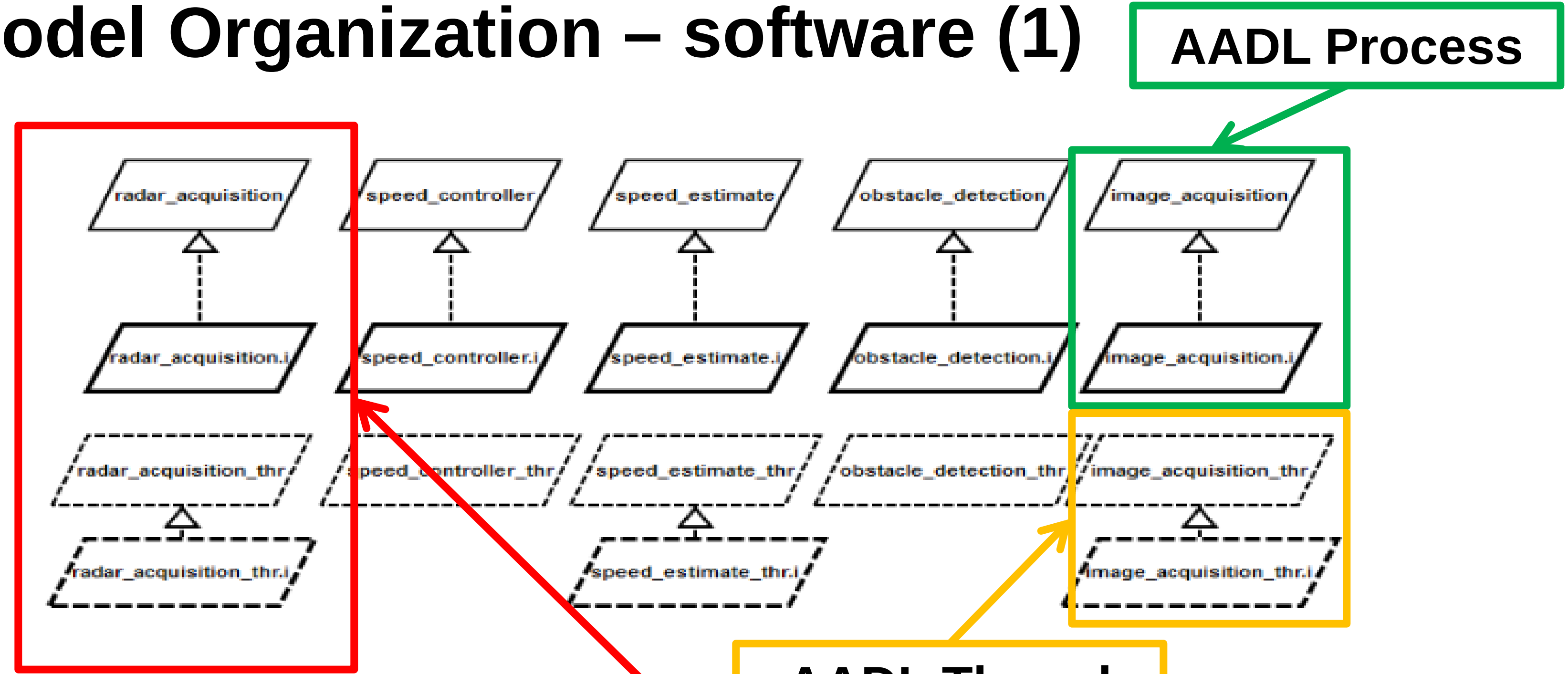
```
processor ecu_can_one_connector extends ecu
features
    socket : requires bus access can;
end ecu_can_one_connector;
```

```
bus can
properties
    Latency => 1 Ms .. 1 Ms;
    SEI::BandWidthCapacity => 500000.0 bitsps;
    Transmission_Time => [ Fixed => 10 ms .. 30ms;
                          PerByte => 1 us .. 10 us; ];
end can;
```

Processor extension, specify bus connections
Share properties of inherited component

Timing information
(latency analysis)

Model Organization – software (1)



One software function = 1 AADL process + 1 AADL thread

Model Organization – software – textual notation (1)

```
process radar_acquisition
features
    obstacle_distance : in data port speed_regulation::icd::distance;
    obstacle_detected : out data port speed_regulation::icd::boolean;
flows
    f0 : flow path obstacle_distance -> obstacle_detected;
annex EMV2 {**
    use types     speed_regulation::error_library;
    use behavior    speed_regulation::error_library::simple;

    error propagations
        obstacle_distance : in propagation {NoValue,InvalidValue};
        obstacle_detected : out propagation {NoValue,InvalidValue};
        processor : in propagation {SoftwareFailure, HardwareFailure};
    flows
        ef0 : error path obstacle_distance{NoValue} -> obstacle_detected{NoValue};
        ef1 : error path obstacle_distance{NoValue} -> obstacle_detected{InvalidValue};
        ef3 : error path obstacle_distance{InvalidValue} -> obstacle_detected{InvalidValue};
        ef2 : error path processor{HardwareFailure,SoftwareFailure} -> obstacle_detected{NoValue};
    end propagations;

    component error behavior
    transitions
        t0 : Operational -[processor{SoftwareFailure}]-> Failed;
        t1 : Operational -[processor{HardwareFailure}]-> Failed;
        t2 : Failed -[processor{NoError}]-> Operational;
    propagations
        p1 : Failed -[]-> obstacle_detected{NoValue};
    end component;
**};
end radar_acquisition;
```

Component type

Communication interfaces

Data flow specification
(latency analysis)

Error specification
(safety analyses)

Subcomponents
and connections

```
process implementation radar_acquisition.i
subcomponents
    thr : thread radar_acquisition_thr.i;
connections
    c0 : port obstacle_distance -> thr.obstacle_distance;
    c1 : port thr.obstacle_detected -> obstacle_detected;
flows
    f0 : flow path obstacle_distance -> c0 -> thr.f0 -> c1 -> obstacle_detected;
end radar_acquisition.i;
```

Component implementation

Model Organization – software – textual notation (2)

```
thread radar_acquisition_thr
features
    obstacle_distance : in data port speed_regulation::icd::distance;
    obstacle_detected : out data port speed_regulation::icd::boolean;
flows
    f0 : flow path obstacle_distance -> obstacle_detected;
properties
    Dispatch_Protocol => Periodic;
    Period            => 10ms;
    sei::mipsbudget   => 4.0 mips;
end radar_acquisition_thr;
```

Data flow
(latency analysis)

Time information
(latency analysis)

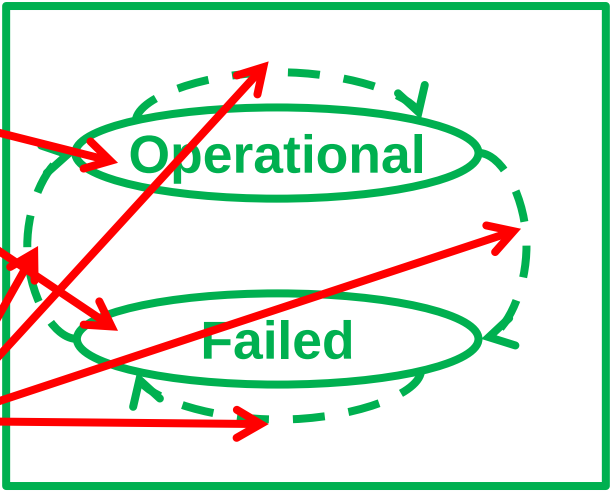
Resource Budgets
(resource allocation analysis)

Model Organization – safety specification

Error types that could be raised

```
package speed_regulation::error_library
public
annex EMV2 {**
    error types
        NoPower                        : type;
        ValueEr
        NoValue                                 alueError;
        Invalid                                 ueError;
        Hardwar
        SoftwareFailure                : type;
    end types;

    error behavior simple
    states
        Operational : initial state;
        Failed      : state;
    end
**};
end spee
```

Error states



Component-specific error transitions
(to be added on a component-basis)

Reusable error state machines
to be attached to components

Model Organization – define error flows – error source

```
device camera
features
    picture : out data port speed_regulation::icd::picture;
flows
    f0 : flow source picture;
properties
    Period => 200ms;
annex EMV2 {**
    use types speed_regulation::error_library;

    error propagations
        picture : out propagation {NoValue};
    flows
        ef0 : error source picture{NoValue};
    end propagations;
**};
end camera;
```

Reuse predefined types

Define error types propagated on component interfaces

Define the the error sources, what interfaces initiates an error flow

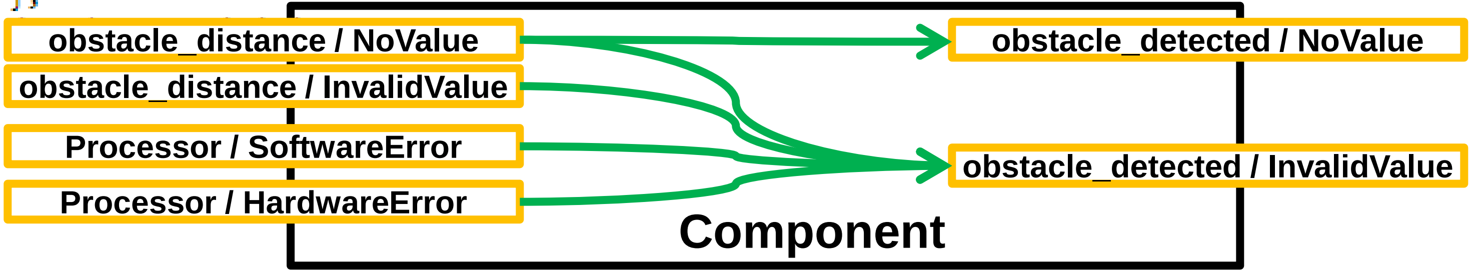
Model Organization – define error flows – error path

Reuse predefined types and behavior

Define error types propagated on component interfaces

Define the propagations flows

```
annex EMV2 {**
    use types       speed_regulation::error_libr
    use behavior    speed_regulation::error_library::simple;

    error propagations
        obstacle_distance : in propagation {NoValue,InvalidValue};
        obstacle_detected : out propagation {NoValue,InvalidValue};
        processor : in propagation {SoftwareFailure, HardwareFailure};
    flows
        ef0 : error path obstacle_distance{NoValue} -> obstacle_detected{NoValue};
        ef1 : error path obstacle_distance{NoValue} -> obstacle_detected{InvalidValue};
        ef3 : error path obstacle_distance{InvalidValue} -> obstacle_detected{InvalidValue};
        ef2 : error path processor{HardwareFailure,                                   e};
    end propagations;
**};
```



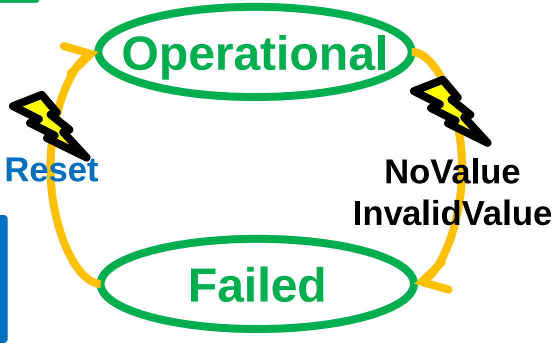
Model Organization – error sink & define component error behavior

```
device warning_device
features
    warning : in data port speed_regulation::icd::boolean;
flows
    f0 : flow sink warning;
properties
    Period => 500ms;
annex EMV2 {**
    use types        speed_regulation::error_library;
    use behavior     speed_regulation::error_library::simple;

    error propagations
        warning : in propagation {NoValue,InvalidValue};
    flows
        ef0 : error sink warning{NoValue,InvalidValue};
    end propagations;

    component error behavio
    events
        Reset : recover event;
    transitions
        t0 : Operational -[warning{NoValue}]-> Failed;
        t1 : Operational -[warning{InvalidValue}]-> Failed;
        t2 : Failed -[Reset]-> Operational;
    end component;
**};
end warning_device;
```

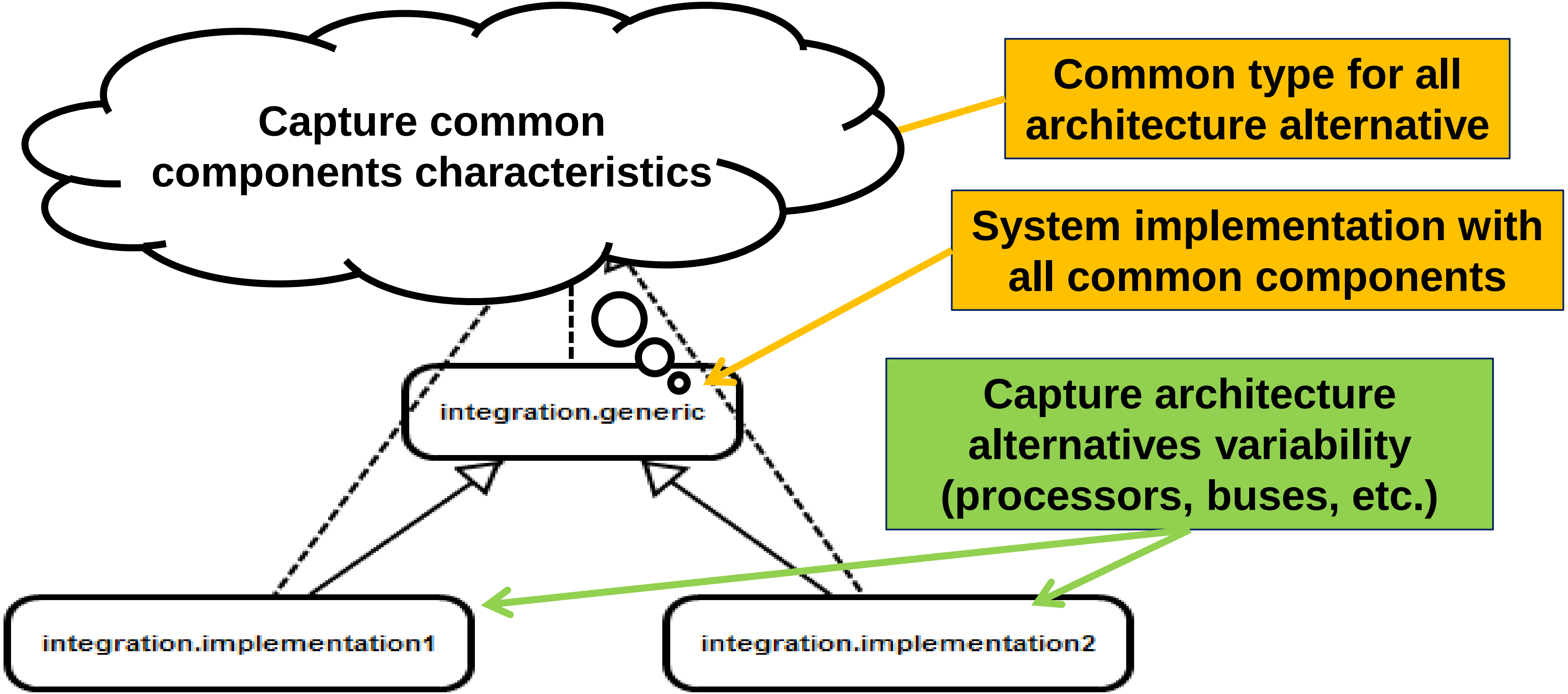
Use predefined error types and component behavior

Define component-specific error events

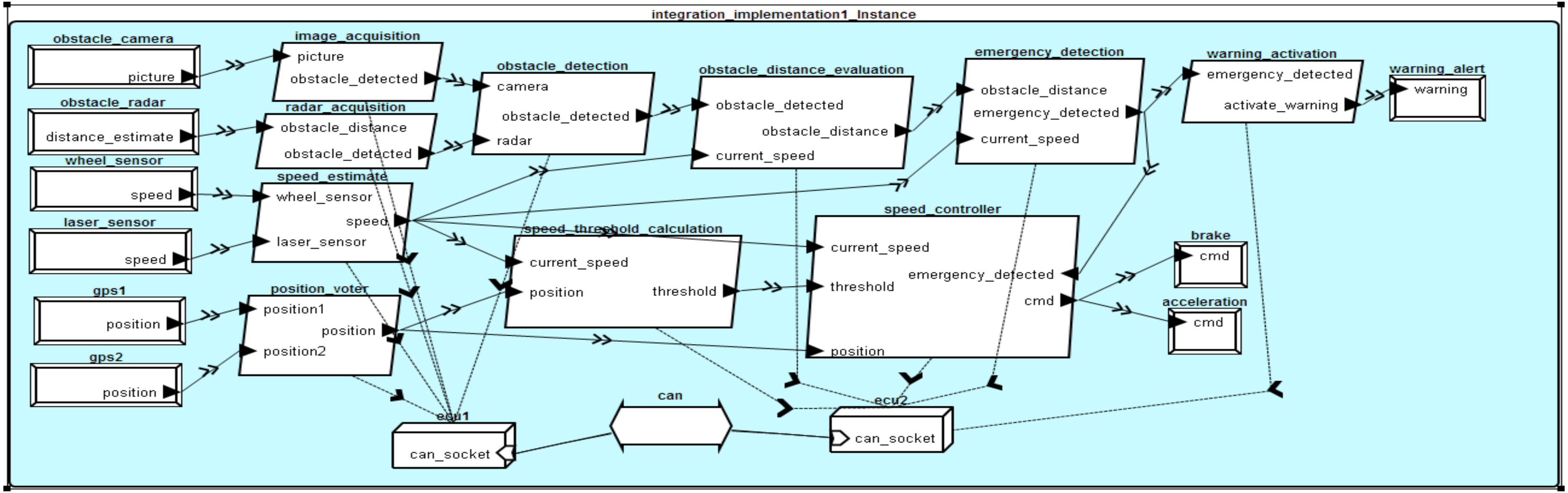
Component-specific error transitions



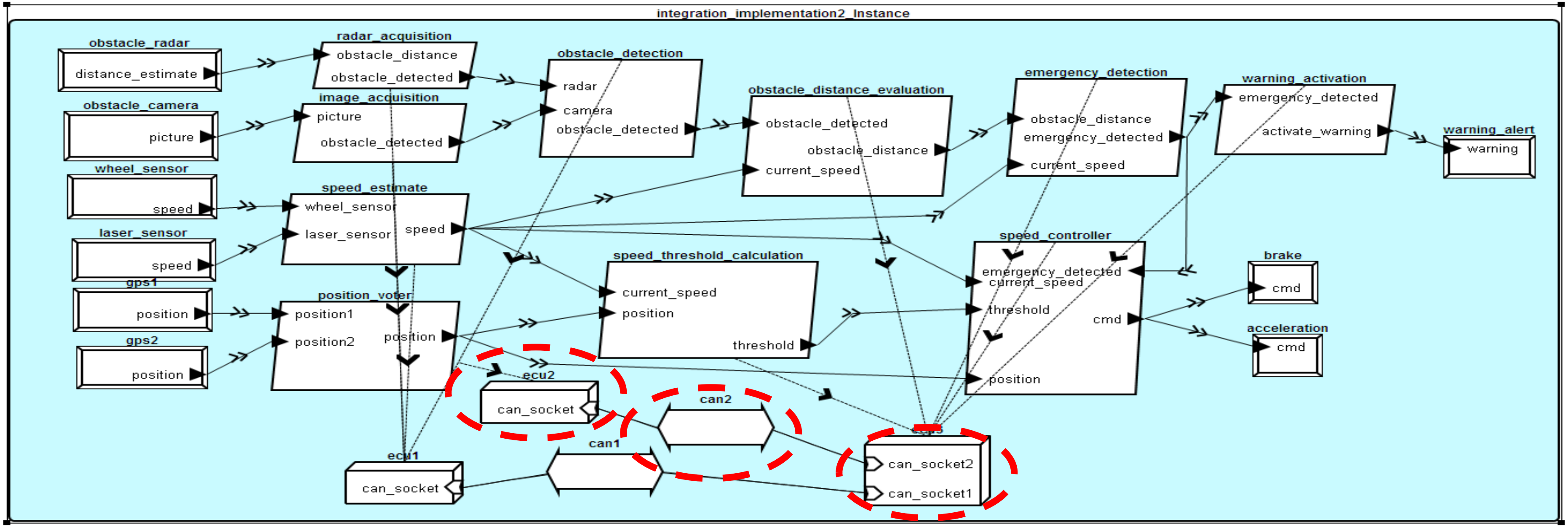
Model Organization – architecture alternatives



Architecture Alternative 1: model instance



Architecture Alternative 2: model instance



Variability Factors with Alternative 1

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

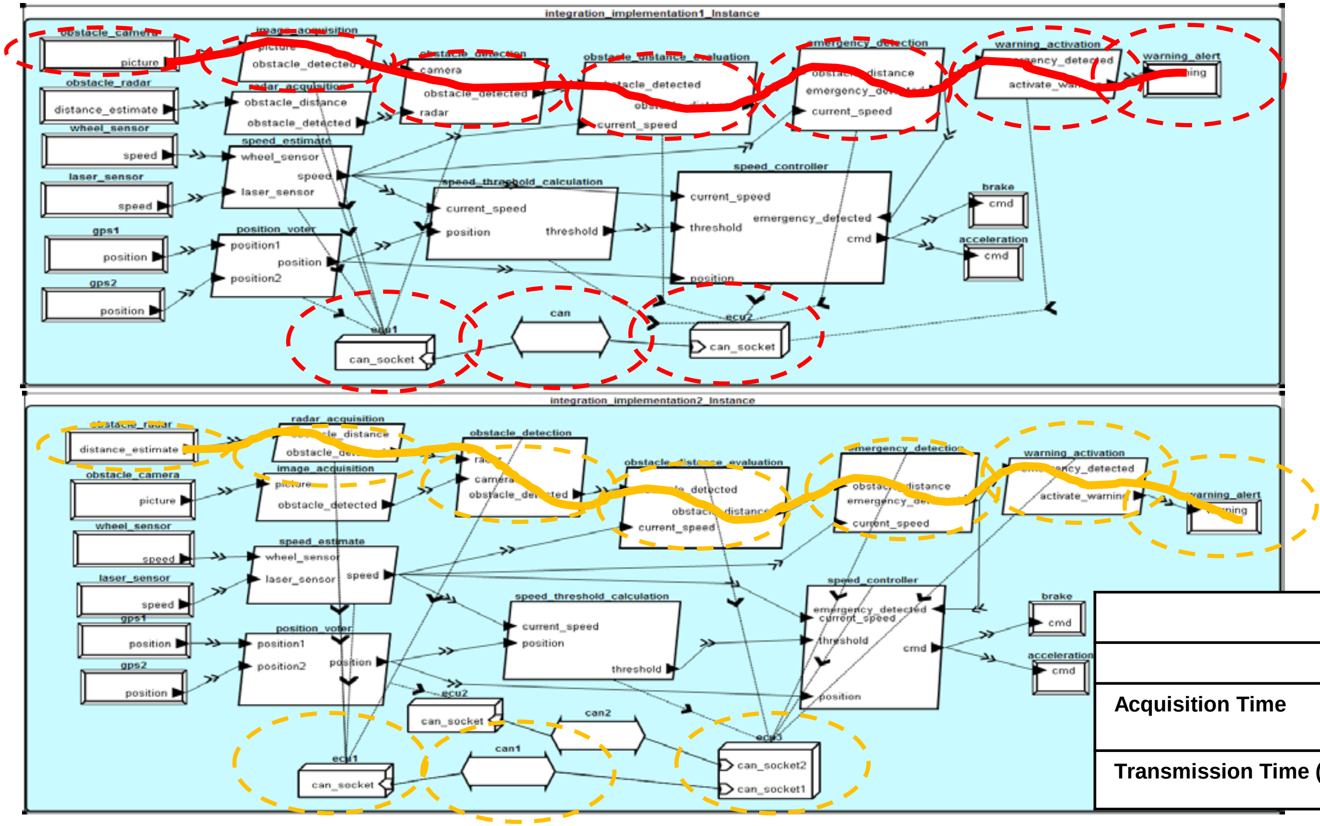
System Overview

AADL model description

Architecture Analysis

Conclusion

Latency Analysis, principles



Potential impact on latency

Bus characteristics		
	Alternative1	Alternative2
Acquisition Time	10 to 30 ms	200 to 500 ms
Transmission Time (/B)	1 to 10us	2 to 5 ms

Latency Analysis, results

Architecture Alternative 1

flow	model element	name	deadline or conn delay	total	expected
f0: End to End Latency report					
f0 (Synchronous)	device	obstacle_camera:f0	200.0 ms	200.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_camera.pictur	0.0 us	200.0 ms	900.0 ms
f0 (Synchronous)	thread	image_acquisition.thr:f	50.0 ms	250.0 ms	900.0 ms
f0 (Synchronous)	Connection	image_acquisition.thr.c	0.0 us	250.0 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_detection.thr:	100.0 ms	350.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_detection.thr.	30.00125 ms	380.00125 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_distance_eval	10.0 ms	390.00125 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_distance_eval	0.0 us	390.00125 ms	900.0 ms
f0 (Synchronous)	thread	emergency_detection.t	4.0 ms	394.00125 ms	900.0 ms
f0 (Synchronous)	Connection	emergency_detection.t	0.0 us	394.00125 ms	900.0 ms
f0 (Synchronous)	thread	warning_activation.thr:	2.0 ms	396.00125 ms	900.0 ms
f0 (Synchronous)	Connection	warning_activation.thr.	0.0 us	396.00125 ms	900.0 ms
f0 (Synchronous)	device	warning_alert:f0	500.0 ms	896.00125 ms	900.0 ms
f0 (Synchronous)	Total		0.0 us	896.00125 ms	900.0 ms

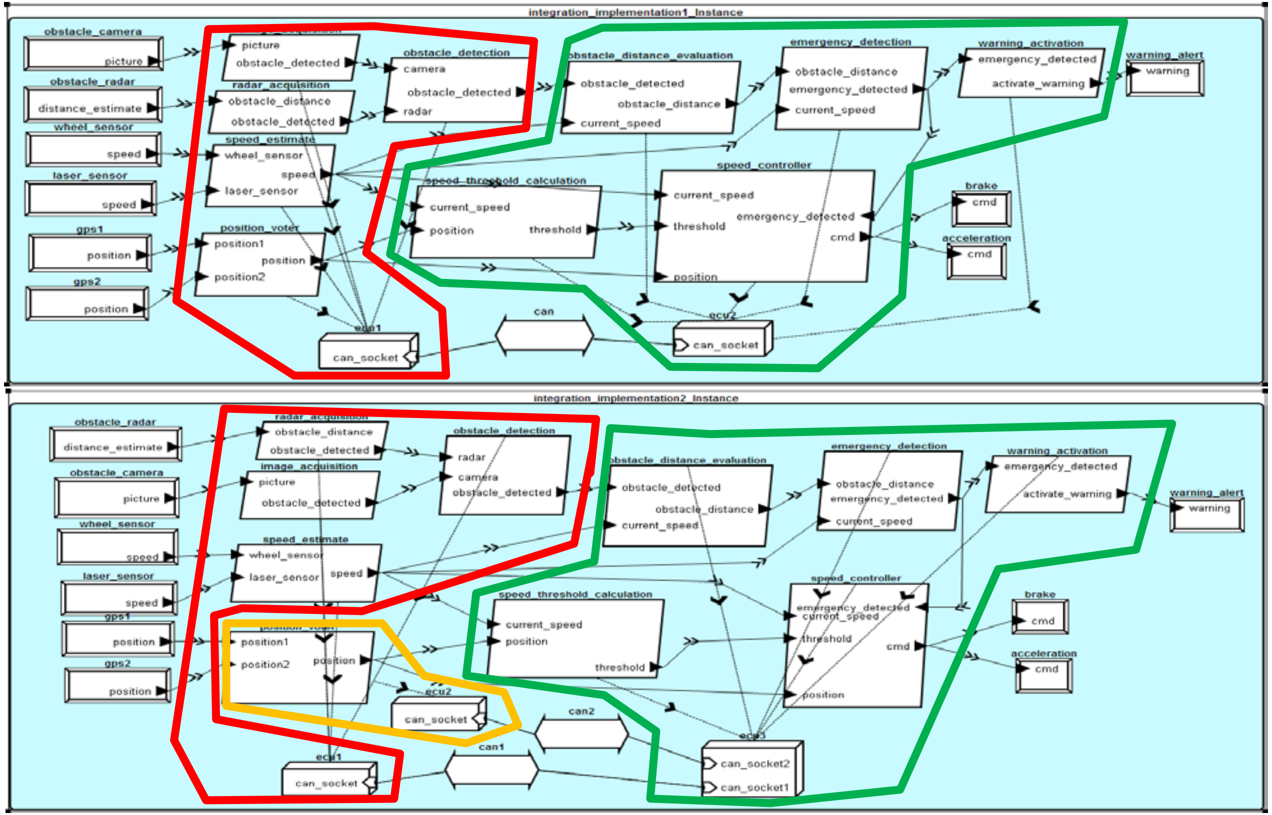
f0: End-to-end flow f0 calculated latency (Synchronous) 896.00125 ms is less than expected latency 900.0 ms

Architecture Alternative 2

flow	model elemer	name	deadline or conr	total	expected
f0: End to End Latency report					
f0 (Synchronous)	device	obstacle_camera:f0	200.0 ms	200.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_camera.picture -	0.0 us	200.0 ms	900.0 ms
f0 (Synchronous)	thread	image_acquisition.thr:f0	50.0 ms	250.0 ms	900.0 ms
f0 (Synchronous)	Connection	image_acquisition.thr.obs	0.0 us	250.0 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_detection.thr:f0	100.0 ms	350.0 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_detection.thr.ob	100.00625 ms	450.00625 ms	900.0 ms
f0 (Synchronous)	thread	obstacle_distance_evalua	10.0 ms	460.00625 ms	900.0 ms
f0 (Synchronous)	Connection	obstacle_distance_evalua	0.0 us	460.00625 ms	900.0 ms
f0 (Synchronous)	thread	emergency_detection.thr	4.0 ms	464.00625 ms	900.0 ms
f0 (Synchronous)	Connection	emergency_detection.thr	0.0 us	464.00625 ms	900.0 ms
f0 (Synchronous)	thread	warning_activation.thr:f0	2.0 ms	466.00625 ms	900.0 ms
f0 (Synchronous)	Connection	warning_activation.thr.ac	0.0 us	466.00625 ms	900.0 ms
f0 (Synchronous)	device	warning_alert:f0	500.0 ms	966.00625 ms	900.0 ms
f0 (Synchronous)	Total		0.0 us	966.00625 ms	900.0 ms

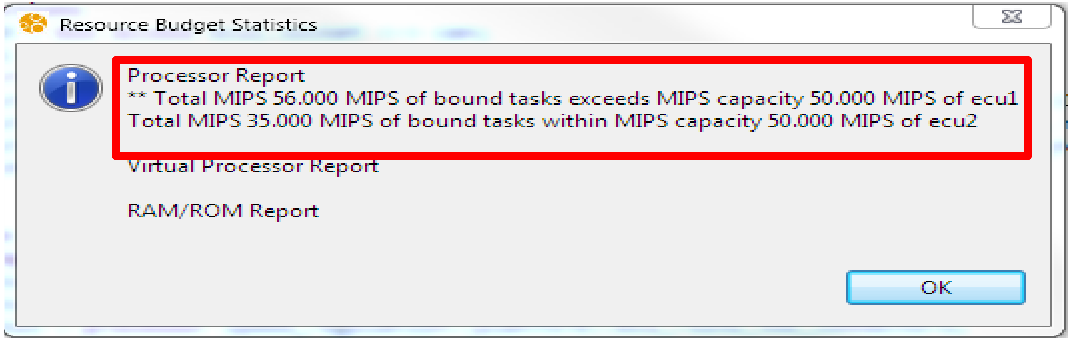
ERROR: f0: End-to-end flow f0 calculated latency (Synchronous) 966.00625 ms exceeds expected latency 900.0 ms

Resources Allocation Analysis, principles

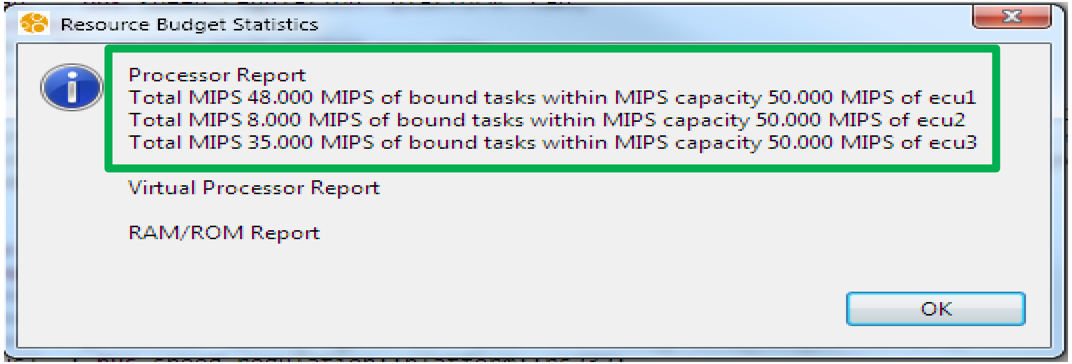


Resources Allocation Analysis, results

Architecture Alternative 1



Architecture Alternative 2



Safety Analyses Overview

Functional Hazard Analysis (FHA)

Failures inventory with description, classification, etc.

Fault-Tree Analysis (FTA)

Dependencies between errors event and failure modes

Fault-Impact Analysis

Error propagations from an error source to impacted component

Need to combine analyses

Connect results to see impact on critical components

Safety Analysis, FHA, results

Architecture Alternative 1: 15 errors contributors 

Architecture Alternative 2: 17 errors contributors 

Difference stems from additional platform components (ecu)

Have to consider criticality of fault impacts

Safety Analysis, FTA results

Architecture Alternative 1: 15 errors contributors

Architecture Alternative 2: 17 errors contributors

Difference stems from additional platform components (ecu)

Have to consider criticality of fault impacts

Safety Analysis, Fault Impact, results

Architecture Alternative 1 & 2: 443 error paths

Use the same paths

The additional ECU in alternative 2 covers path from ecu2 in Alternative 1

Impact on components criticality

Defect on the additional bus in Architecture 2 impact low-critical functions

Isolate defect from low-critical functions to affect high-critical

Analysis Summary

	Architecture 1	Architecture 2
Latency	✓	✗
Resources Budgets	✗	✓
Safety	✗	✓
Cost	✓	✗

What is the *“best”* architecture?

Agenda

Introduction on Model-Based Engineering

Presentation of the Case Study

System Overview

AADL model description

Architecture Analysis

Conclusion

Conclusions

Safety-Critical Systems Development issues is not a fatality

- Late detection of errors is no longer possible
- Need for new methods and tools

AADL supports Architecture Study and Reasoning

- Evaluate quality among several architectures
- Ease decision making between different architecture variations
- Analysis of Architectural change on the whole system

User-friendly and open-source workbench

- Graphical Notation
- Interface with other Open-Source Tools

Useful Resources

AADL wiki – http://www.aadl.info/wiki

Model-Based Engineering with AADL book

SEI blog post series http://blog.sei.cmu.edu

Mailing-List
see. https://wiki.sei.cmu.edu/aadl/index.php/Mailing_List

Questions & Contact

Dr. Julien Delange
Member of the Technical Staff
Architecture Practice
Telephone: +1 412-268-9652
Email: info@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257